INTEGRATED CIRCUITS

SPECIAL TOOLS AND CHIPS MAKE FUZZY LOGIC SIMPLE

GARY LEGG, Senior Technical Editor

Inexpensive software tools make fuzzy logic easy to learn and apply. Dedicated fuzzy processors and enhanced microcontrollers make applications run fast.

IF YOU WOULD RATHER DERIVE COMPLICATED mathematical equations than approach a problem from an intuitive point of view, then you probably won't be interested in fuzzy logic. Likewise, if you have unlimited time and money to spend on design tasks, then fuzzy logic may not appeal to you. And if you don't mind your designs being expensive to produce and hard to modify, then you won't want to waste your time on something that sounds so, well, fuzzy.

But if you live in the *real* world, you need to know about fuzzy logic. Although it may sound like a laughable oxymoron, fuzzy logic is quickly gaining respect as a technically viable and cost-effective discipline, especially for embedded control. Fuzzy logic is also easy to learn and apply, and a handful of companies have products that can help you get started.

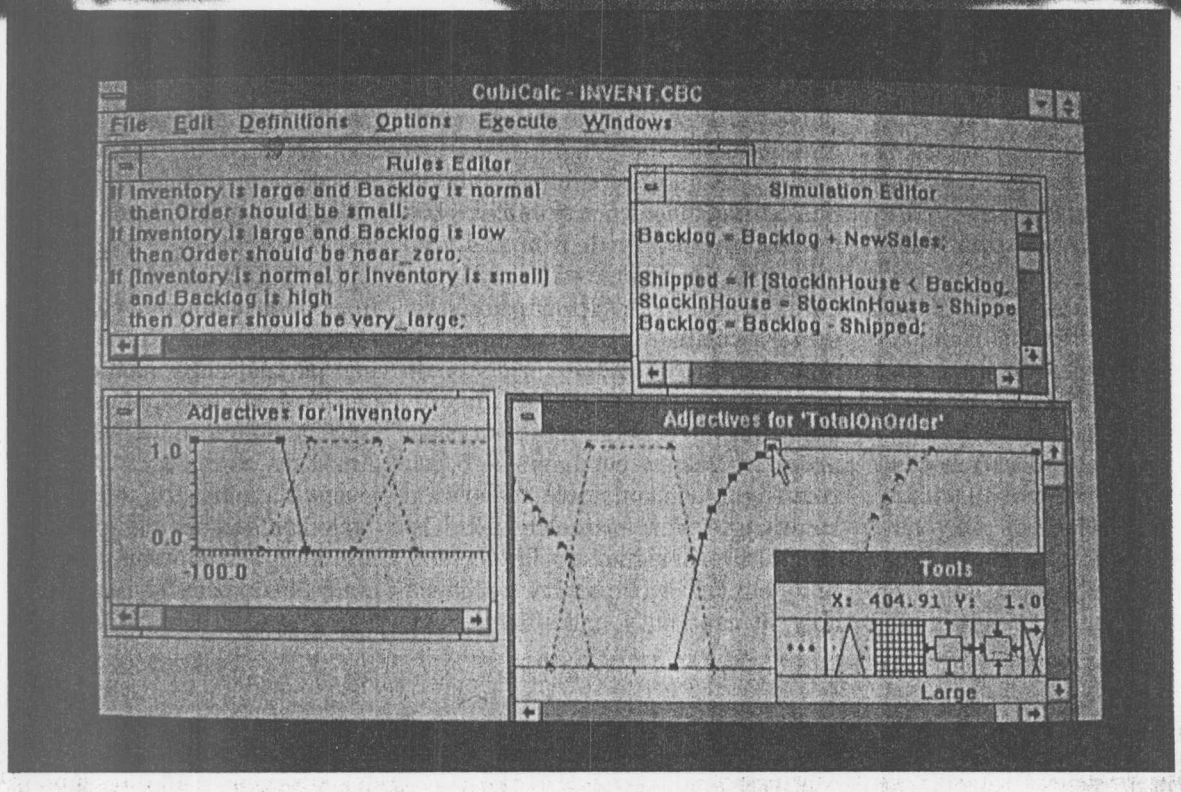


Fig 1—Fuzzy-logic software tools help you write controller rules and sketch membership functions. Here, the graphics editor in Hyperlogic's Cubicalc helps specify an output membership function (lower right quarter of display).

Fuzzy-logic products for designers include both software and hardware. The software ranges from freeware fuzzy-logic inference kernels for microcontrollers to computer-aided software engineering (CASE)-like packages that minimize application programming. The most elaborate packages provide help in all the steps of designing, simulating, and testing a fuzzy controller.

Fuzzy-logic hardware includes dedicated fuzzy-logic processors and coprocessors, plus computer boards that use those chips. You can get fuzzy-logic educational kits, which include some software, for as little as $195; for $400 to $600, you can get both software and a fuzzy-logic board that plugs into a PC. Even full-scale professional software tools for fuzzy logic are fairly inexpensive, ranging in price from a few hundred dollars to a few thousand.

Fuzzy-logic development tools simplify controller design by capitalizing on fuzzy logic's intuitive, common-sense appeal (see **box**, "Basic fuzzy-

logic design"). With some of the tools, you can show how you want a controller's outputs to relate to its inputs simply by drawing sketches and writing plain-language rules (**Fig 1**). Then, through simulation, the tools produce graphical displays of your controller's behavior (**Fig 2**) and let you alter that behavior by simply altering your sketches and rules.

Fuzzy-logic tools are especially valuable for prototyping. You don't have to be a programmer to use the tools; you just have to know how your system should behave. However, the tools will also help you produce code for actually implementing your fuzzy-logic system. Some generate C code; others produce assembly language. Of the assembly-language versions, about half produce code for familiar 8-bit microcontrollers; code from the others runs on dedicated fuzzy-logic chips. (For brief descriptions of some available tools, see **box**, "A fuzzy-logic toolbox.")

Although you can implement fuzzy logic on a general-purpose processor, a few companies—Togai Infralogic, Omron, and American Neuralogix, for example—sell ICs specifically designed to implement fuzzy logic. The chips vary considerably in capability and price, from around $10 for a simple fuzzy coprocessor to about $200 for a standalone, RISC-architecture fuzzy processor.

Major semiconductor companies are also getting interested in fuzzy logic. Several have alliances with fuzzy-logic companies and are planning future fuzzy products. Existing partnerships include Motorola and Aptronix, Siemens and Togai, NEC and Omron, and Samsung and American Neuralogix. SGS-Thomson is also planning fuzzy-logic products.

Dedicated fuzzy chips are useful for complex control systems, but many fuzzy applications don't require special hardware. Because

Basic fuzzy-logic design

Fuzzy logic works as humans do in that it easily processes imprecise (fuzzy) variables such as "warm" or "fast" or "slight throttle" instead of insisting on needless numerical precision. A primary underlying principle of fuzzy logic is that precision is often unnecessary and even counterproductive (**Ref 1**). By relying less on precision and more on a human designer's seat-of-the-pants intuition about how something should work, fuzzy-logic controllers can often be simpler, cheaper, and more reliable than traditional controllers.

Consider, for example, how you steer a car. You steer slightly to the left when your car begins to drift to the right. You don't, and probably couldn't, make steering decisions based on numerical inputs about your car's heading. Fuzzy logic works much the same way.

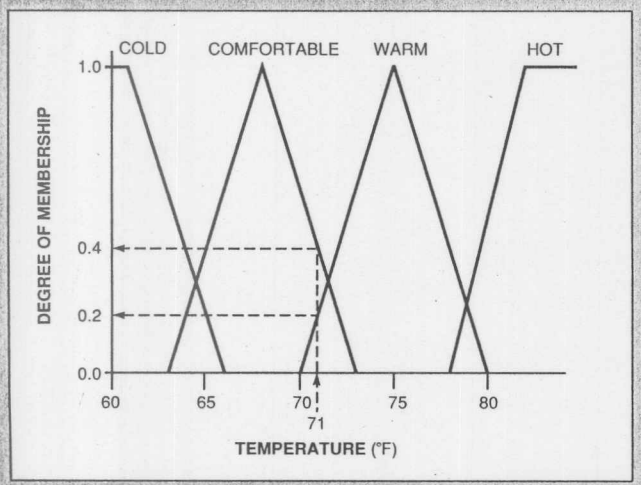


Fig A—Fuzzification is the process of converting crisp (numerical) values to fuzzy values. Here, the element 71°F has degree of membership 0.4 in fuzzy set Comfortable and 0.2 in the fuzzy set Warm.

Hardware systems necessarily receive numerical inputs, however, so a fuzzy controller must "fuzzify" those inputs in order to use them. For example, a controller might convert a certain measured temperature to a fuzzy variable called Warm. The controller then applies the fuzzified inputs to an inference mechanism, consisting of if-then rules, to determine what actions to invoke. An example rule might be, "If Temperature is Warm, then make FanSpeed Medium." Finally, the controller must "defuzzify" actions in order to apply them to an actual device. For example, it might convert a medium fan speed to a voltage that will turn the fan at 200 rpm.

The design of a fuzzy controller follows the three steps of fuzzification, inference, and defuzzification. In each step, you just use your common sense and intuition.

In fuzzification, a controller applies numerical (crisp) inputs to simple functions that define crisp-to-fuzzy transformations (**Fig A**). You define and sketch these functions as part of your design activity. Each function is merely one that seems reasonable to you. Most functions have the shape of a triangle or a trapezoid. If a function turns out not to be adequate, as determined by testing a fuzzy-controller design, it is easy to modify.

Each function actually defines a fuzzy set. The theory of fuzzy sets is different from that of conventional sets. In fuzzy logic, an element may be *partially* a member of a fuzzy set. We say that it has a certain "degree

fuzzy logic is not very demanding of computing power, ordinary 8-bit microcontrollers are adequate for some tasks that might require a 32-bit processor for a conventional design. As a result, fuzzy products are often considerably cheaper to produce than conventional products.

Ordinary microcontrollers are usually adequate for fuzzy-logic control tasks as long as the system sampling rate isn't too high. According to fuzzy-logic consultant David Brubaker of the Huntington Group (Menlo Park, CA), an 8-bit microcontroller will probably work well if the sampling interval is a half second or greater.

Motorola's fuzzy-logic strategy is three-pronged: existing microcontrollers, enhancements to those microcontrollers, and dedicated fuzzy processors. The enhancements will speed up the fuzzy functions that demand extra computations; dedicated chips will be for specific applications. Steve Marsh, director of strategic operations for Motorola's Austin microcontroller division, says dedicated fuzzy processors make sense only for applications, such as graphics, that don't already use conventional microcontrollers. Fuzzy processors will not replace microcontrollers, Marsh says, but augment them.

Whether you design fuzzy logic using a dedicated chip or a microcontroller, your approach will not follow convention. In a conventional control system of at least moderate complexity, you describe the desired system behavior with equations or look-up tables. Often, though, equations are difficult or even impossible to express, and you may have to derive entirely new equations to make a relatively simple system modification. Look-up tables have disadvantages, too. They can be long, using scarce microcontroller memory, and they can result in jerky system response as a conventional controller steps

of membership" ranging from 0 to 1, inclusive. Thus, the functions that you define are called input-membership functions.

Fuzzy logic also lets elements be partially in one set and partially in another. In **Fig A**, the temperature 71°F has a degree of membership of 0.4 in the set Comfortable and a degree of membership of 0.2 in the set Warm. The overlap of membership functions corresponds to human notions; we say that a certain temperature, for example, corresponds both to "fairly comfortable" and to "slightly warm."

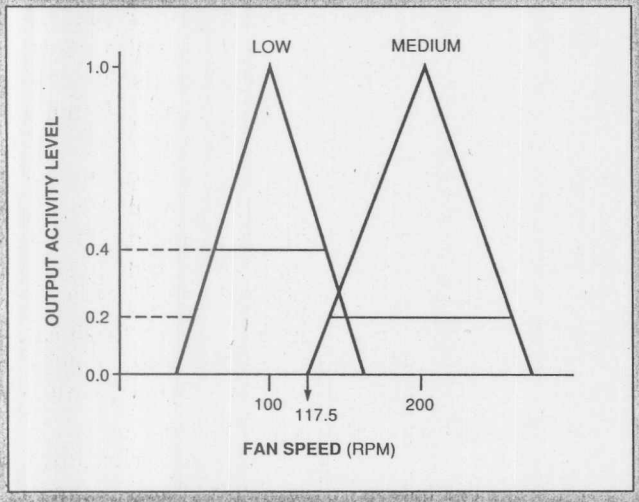


Fig B—Defuzzification of fuzzy values results in crisp values that can apply to actual devices. Here, the fuzzy values Low and Medium (representing fan speeds) combine to form an intermediate value in numerical form. The value is closer to Low than to Medium because the condition corresponding to Low is more nearly true than the condition corresponding to Medium.

Because an element can be a partial member of a fuzzy set, a fuzzy controller can take partial action based on that membership. A fuzzy controller can also combine the actions that are based on membership (full or partial) in different fuzzy sets. For the situation corresponding to **Fig A**, a controller would ideally place twice as much emphasis on the action associated with Comfortable (for example, "make FanSpeed Low") as on the action associated with Warm (for example, "make FanSpeed Medium"). The result (**Fig B**) would be a fan speed between low and medium, but closer to low. A human would probably do much the same thing.

Defuzzification must accompany the combining of actions for you to obtain a crisp value that you can apply to an actual device. For example, you would need to convert a fuzzy fan speed between low and medium to an actual rotational speed, or perhaps an appropriate motor voltage. Combination and defuzzification occur via output membership functions, which you define as you do input membership functions, by sketching something that seems reasonable.

Different methods exist for combining actions. One common method, illustrated in **Fig B**, computes the centroid of all applicable output membership functions. Many fuzzy controllers work well, however, by invoking only the one action that the inference mechanism suggests is the most appropriate. This robustness is typical of fuzzy controllers and is a main reason why fuzzy logic is gaining so much favor.

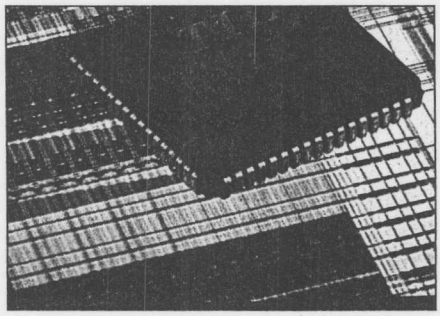
from one table value to another.

In a fuzzy control system, you describe the desired behavior in simple rules that are based on your practical, intuitive understanding of the problem. With some of the CASE-like tools, you can express rules in ordinary language; a typical rule would be something like, "If Temperature is Hot, then make FanSpeed High." A simple fuzzy controller might require less than ten rules; a complex one might need 30 or 40.

A fuzzy controller applies the rules to system inputs to determine appropriate system actions. First, however, you must convert those inputs from "crisp" numbers (measured temperatures, for example) to "fuzzy" variables that your fuzzy controller can use. In fuzzy-logic jargon, this process is called fuzzification.

Fuzzification of crisp inputs occurs via input-membership functions, which you define. For example, you could use a simplified bell curve centered at 68°F to define "Comfortable."

Most membership functions, in fact, have simple, geometric



The FC110 fuzzy processor from Togai Infralogic uses a RISC architecture to speed computations.

shapes. Triangle shapes are a crude approximation to bell curves, and they work well for a majority of fuzzy-logic control applications. A step up in complexity is the trapezoid, essentially a triangle with its top lopped off. According to Earl Cox, a Chappaqua, NY-based consultant in fuzzy-logic applications, triangles and trapezoids are adequate for about 90% of applications. (See the two Design Features on fuzzy logic in the June 18, 1992, EDN for more-detailed examples.)

When you design a fuzzy controller, you specify the shapes of membership functions by coding a few point coordinates or by using a graphics editor in one of the available fuzzy-logic software tools. Tools with graphics editors include Cubicalc ($495) from Hyperlogic, TILShell ($2300 to $3300) from Togai Infralogic, and Fide (pronounced fee DAY, $1495) from Aptronix. A limited-capability version of Fide is included in Motorola's $195 fuzzy-logic education package.

Some fuzzy-logic chips allow membership functions of arbitrary shapes; others limit functions to only a few shapes or even to triangles only. Togai's $200 FC110 fuzzy processor, for example, allows user-defined arbitrary shapes; Omron's FP-3000 fuzzy coprocessor, which sells for $75 to $100, allows four different shapes. The $10 Neuralogix NLX230 allows only triangles.

In microcontroller-based fuzzy applications, membership functions face practical limits imposed by processing requirements. To keep computation time within reason, the functions must be relatively simple. You can partially avoid the computation bottleneck by choosing an appropriate microcontroller, however. Motorola's 68332, for example, has a table-interpolate instruction that speeds up calculations involving pairs of membership-function points.

Defuzzification of system actions occurs via output membership functions, which you define as you do input membership functions. This process, to be discussed later, is more complicated than fuzzification, however, because it involves combining the actions that result from multiple rules.

The process of inferring fuzzy actions by applying fuzzy inputs to a rule base is called, appropriately, fuzzy inference. Fuzzy-logic implementations use two common inference mechanisms—denoted max-min (or min-max) and max-dot (**Ref 1**)—but you probably won't be concerned with their differences until you're well past the novice stage

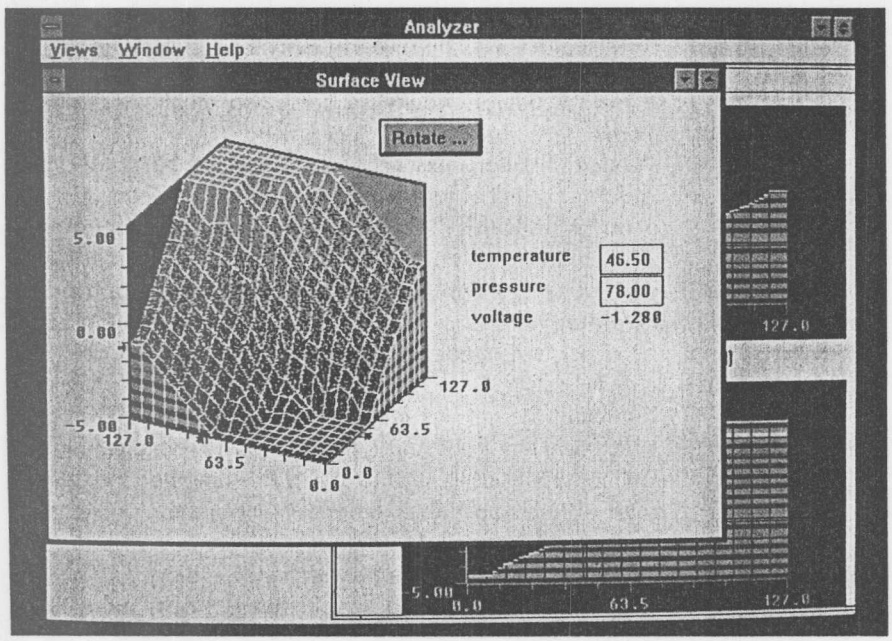


Fig 2—Aptronix's Fide provides three different displays of input/output relationships. In this one, a 3-D surface shows the system output as a function of two inputs. Others reveal the same information in contour displays and cross-section views.

in fuzzy design. Despite their differences, Brubaker notes, both methods operate basically the same.

Partial truth and consequence

The fuzzy inference process determines which fuzzy rules "fire," or are true. Unlike the rules of conventional logic, however, which fire completely or not at all, fuzzy rules fire only to the degree that their antecedents (the "if" parts) are true. This degree of truth determines an output activation level, or the degree to which the rules' consequents (the "then" parts) will be invoked. You might say that if a condition "sort of" exists, then a corresponding action is "sort of" invoked.

After a fuzzy controller determines what actions to invoke and at what levels, it must defuzzify the fuzzy actions so that they produce an appropriate crisp action on an actual device. Ideally, the controller also combines actions that result from different rules. This combining of actions, each of which may be a "partial" action, contributes to fuzzy logic's robustness, reliability, and similarity to human reasoning.

Fuzzy-logic implementations use several methods for combination and defuzzification (**Ref 1**), but some fuzzy systems don't combine actions at all; they invoke only the one action corresponding to the rule that is "most true." Surprisingly, such systems can be very accurate and reliable.

The commonly used method that is best for combining actions computes the centroid, or center of gravity, for the combination of all clipped or scaled output membership functions. The centroid method requires a great deal of computa-

A fuzzy-logic toolbox

Software tools are a valuable aid in designing fuzzy-logic applications. They're also a bargain. Full-featured tools range in price from a few hundred dollars to a few thousand. Some software is even free.

The freeware is from Motorola. You can download it from Motorola's computer bulletin board; the phone number is (512) 891-3733. The two components of the freeware are a fuzzy-logic kernel and software that helps you create membership functions. You can use the freeware for developing applications on 68HC05 and 68HC11 microcontrollers.

Motorola also offers fuzzy-logic educational kits. The basic kit, for $195, includes a demo version of Fide, the professional fuzzy-logic tool from Aptronix. For $600, you also get a hardware emulator for your choice of microcontroller.

You can also get a complete version of Fide either from Motorola or Aptronix. The latest version, which sells for $1495, includes a graphical editor for membership functions. Other key features are simulation, analysis, and trace capabilities. Fide also includes what Aptronix calls a "composer" for merging fuzzy code with other system code. In addition to generating assembly code for Motorola's 68HC05 and 68HC11 microcontrollers, Fide also produces C code. A future version, due later this year, will support Motorola's 68HC16 and 68300 families and 56000 DSP family.

Cubicalc ($495) from Hyperlogic is a capable prototyping tool. Its features include graphical editing of membership functions. Cubicalc RTC, a $795 expanded version of Cubicalc, generates C code. RTC requires either a Microsoft C or a Borland Turbo C compiler. The code it produces will run in RAM-based or ROM-based embedded systems and with real-time executives for 8-bit microcontrollers. The RTC package includes binary run-time libraries for 80x86 processors; source code is available if you need to use the libraries on other processors. Hyperlogic also sells Cubicalc RTC in combination with Cubicard, a PC-compatible plug-in card that's useful for control-system applications. The combination package costs $1495.

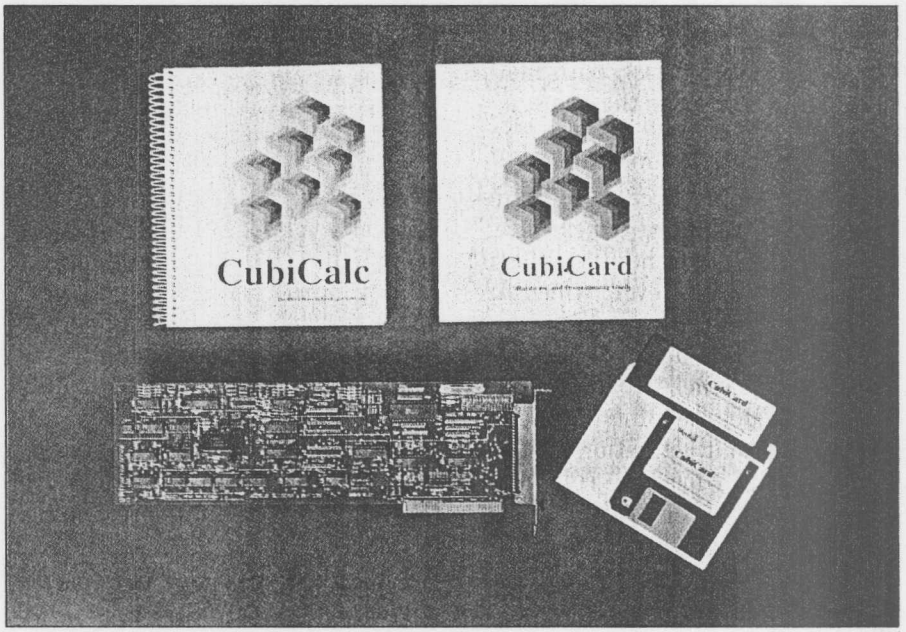
Togai Infralogic's tools are among the most extensive, and expensive, available. The suite of products includes modules for graphical design, graphical analysis, and code development. Complete development-system packages come in several different versions. One generates C code, others produce code for specific processors. Those processors include Togai's own FC110 fuzzy processor; the H8/300, H8/500, and HMCS400 from Hitachi; and Mitsubishi's 37450. Togai also sells an SBus fuzzy-logic accelerator board. Prices for Togai products range from $750 for the FC110 development system to $18,500 for a version of the MicroFPL development system for a specific processor.

Other software tools help you produce code for the tool makers' own fuzzy chips and boards. American Neuralogix, for example, provides some basic software as part of its $395 ADS230 training and development system. The package includes a PC board with a resident NLX230 fuzzy processor.

Omron's FS-10AT software ($695) runs on a PC and produces object code for Omron's FP-3000 fuzzy processor or PC-resident FB-30AT fuzzy-inference board ($1100). These and other fuzzy products from Omron—including various types of modules for programmable-logic controllers—are aimed primarily at industrial control systems.

tion, however, so its use is somewhat restricted. If your application requires the centroid method and is time-critical, you may need a fuzzy processor, such as the Omron FP-3000 or the Togai FC110, that implements the centroid method in silicon.

Some microcontrollers are adept at computing centroids, however. New versions of Motorola's 68HC11, for example, have built-in math coprocessors that can speed up the required multiply/accumulate functions. A coprocessor for the 68HC05 is also in the works. The 68HC16 needs no coprocessor; it has a standard multiply-accumulate instruction.



Hyperlogic's Cubicalc aids development of fuzzy-logic applications. Cubicard is a PC-compatible plug-in card that's useful in control-system applications.

Begin with a prototype

Your initial efforts in designing a fuzzy-logic controller probably won't involve such nitty-gritty hardware details, however. To begin, you'll probably design a prototype; then you'll get it to work properly in an iterative process of simulation and redesign.

Cox observes that building a fuzzy system is really a matter of defining a control surface for the system; the system operates by converging to a solution on that surface. Essentially, Cox says, you sketch out what you believe are the underlying fuzzy sets, or membership functions, associated with every term in your system. Then you write rules that state what should happen when system inputs are members of those fuzzy sets. Finally, you run the system on a simulator, and you tweak the rules and membership functions until your system performs correctly.

Software simulators greatly assist your checkout efforts by showing you the effects of fuzzy-system inputs on system outputs. Aptronix's Fide, for example, provides three different displays of input/output relationships. In one, a 3-D surface shows the system output as a function of two inputs. A second view reveals the same information in a contour display. A third display shows

For more information . . .

For more information on the fuzzy-logic products discussed in this article, circle the appropriate numbers on the Information Retrieval Service card or use EDN's Express Request service. When you contact any of the following manufacturers directly, please let them know you read about their products in EDN.

American Neuralogix Inc
411 Central Park Dr
Sanford, FL 32771
(407) 322-5608
FAX (407) 322-5609
Circle No. 670

Aptronix Inc
2150 N First St
San Jose, CA 95131
(408) 428-1881
FAX (408) 428-1884
Circle No. 671

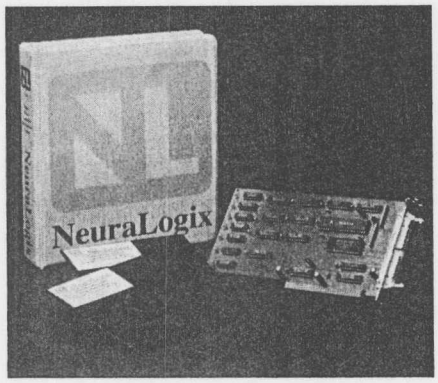
Hyperlogic
1855 E Valley Pkwy
Suite 210
Escondido, CA 92027
(619) 746-2765
FAX (619) 746-4089
Circle No. 672

Motorola Inc
Box 1466
Austin, TX 78767
(512) 891-2840
Circle No. 673

Omron Corp
Shimokaiinji, Nagaokakyo-city
Kyoto 617, Japan
(81) 75-951-5117
FAX (81) 75-952-0411
Circle No.674
In the US:
Omron Electronics Inc
1 E Commerce Dr
Schaumburg, IL 60173
(708) 843-7900
FAX (708) 843-7767
or (708) 843-8568
Circle No. 675

Togai Infralogic Inc
5 Vanderbilt
Irvine, CA 92718
(714) 975-8522
FAX (714) 975-8524
Circle No. 676

VOTE . . .
Please also use the Information Retrieval Service card to rate this article (circle one):
High Interest 473 Medium Interest 474 Low Interest 475



The ADS230 training and development system from American Neuralogix includes software and a PC board with a resident NLX230 fuzzy processor.

system output in two cross-section views, each for a different input.

If a display reveals an output anomaly, you can use a trace feature in Fide to identify the rule or rules at fault. Cubicalc's display capabilities are less extensive, but flexible. Built-in display features include a strip chart (a variable versus time) and a scattergram (one variable against another). However, you can also link Cubicalc to other Windows applications and thus use the plotting tools of other programs—Microsoft's Excel, for example—to display simulation results.

When your fuzzy system works the way it should and you're ready to implement it in program code, software tools can simplify the job. They generate either C code or assembly language from your fuzzy rule base and membership functions. Some also have features to help merge the fuzzy code they generate with other system code.

If you're anticipating developing an actual fuzzy application, you'll eventually have to choose a processor to run it. Whether you choose a dedicated fuzzy processor or a standard microcontroller, an intelligent choice will depend on your having more information than this article can provide. **Ref 1** through **Ref 6** provide good general information, but you'll need some actual practice with fuzzy design to know what hardware is best for certain applications.

Different fuzzy processors, for example, have different capacities for number of rules, number of antecedents per rule, number of consequents per rule, and number of membership functions per input. They also provide different inference methods and allow different types of membership functions. You have to determine your requirements for each of these parameters, and you need experience to do that.

The best advice for getting that experience is, "Just do it." Get one of the educational kits, software packages, or development kits and plunge in. It won't cost much, and you'll probably have fun. In a widely used training example, you can design a fuzzy dog that chases an elusive fuzzy cat. EDN

References

1. Brubaker, David I, "Fuzzy-logic basics: Intuitive rules replace complex math," *EDN*, June 18, 1992, pg 111.

2. Brubaker, David I, and Cedric Sheerer, "Fuzzy-logic system solves control problem," *EDN*, June 18, 1992, pg 121.

3. Cox, Earl, *Selected Topics in Fuzzy Logic and Approximate Reasoning*, a series of private monograms, Chappaqua, NY, 1991.

4. *Fuzzy Logic: A 21st Century Technology*, Publication FUZ-1A, Omron Electronics Inc, Schaumburg, IL, 1991.

5. *Fuzzy Logic from Software to Silicon: A Primer*, Togai Infralogic Inc, Irvine, CA, 1992.

6. Sibigtroth, James M, "Creating Fuzzy Micros," *Embedded Systems Programming*, December 1991.

Acknowledgment

The author is grateful to David Brubaker of the Huntington Group and Earl Cox, a private consultant, for consultation during the preparation of this article. Credit is also due to EDN staff members John Gallant and Steven Leibson for providing helpful comments and suggestions.

Article Interest Quotient
(Circle One)
High 473 Medium 474 Low 475